Application for
UNITED STATES LETTERS PATENT

of

TAKEHIRO SHIMIZU

and

FUMIO ARAKAWA

for

**SIMD OPERATION SYSTEM CAPABLE OF
DESIGNATING PLURAL REGISTERS**

SIMD OPERATION SYSTEM CAPABLE OF DESIGNATING PLURAL
REGISTERS

BACKGROUND OF THE INVENTION

The present invention relates to a register
designating system and in-register data alignment
processing in an SIMD (Single Instruction Multiple Data)
processor, and relates to means for executing in-register
data alignment which may deteriorate a parallel processing
function of SIMD at high speed.

Further, the present invention relates to means for
enabling to operate standard multiply accumulate operation
as a DSP (Digital Signal Processing) instruction while
maintaining parallelism of SIMD and without deteriorating
accuracy.

Japanese Patent Laid-Open No. 124484 has disclosed a
method of executing vector operation such that a plurality
of consecutive registers can be designated by a single
register designating field, when a vector operation
processing which is necessary in three-dimensional graphics
or the like is executed.

Further, as an in-resister data alignment
instruction, there is described various data alignment
instruction which can designate up to four operands in
"AltiVec Programming Interface Manual" of an instruction
set Altivec for multimedia developed by Motorola

Corporation.

Further, with regard to multiply accumulate operation, multiply accumulate instruction is defined, without deteriorating accuracy, which is realized in the form of halving reduced parallelism of SIMD from 4 parallel to 2 parallel by an SH5 architecture developed jointly by Hitachi, Ltd. and ST Microelectronics.

However, according to the vector operation processing disclosed in Japanese Patent Laid-Open No. 124484, a constitution capable of designating only a number of a multiple of 4 is used in designating a plurality of registers and therefore, the constitution is devoid of a degree of freedom. Further, according to the data alignment instruction of Altivec, not only an operating apparatus is large-sized and expensive but also only three source registers can be designated and operation particular to SIMD such as data pack or unpack cannot be executed efficiently. Therefore, the parallelism of SIMD cannot fully be achieved.

SUMMARY OF THE INVENTION

It is a principal object of the invention to provide means with regard to data alignment for maximizing an effect of SIMD instruction.

Further, it is another object of the invention to provide a means for realizing a standard multiply

accumulate operation instruction as a DSP instruction without deteriorating parallelism of SIMD and while maintaining accuracy thereof.

A simple explanation will be given of an outline of representative aspects of the invention disclosed in the application as follows.

According to an aspect of the invention, there is provided a processor including an operation instruction comprising an instruction code and at least one register designating field, wherein the at least one register designating field is capable of designating a plurality of registers having consecutive numbers.

Further, according to another aspect of the invention, there is provided a processor comprising:

a decoder designating a plurality of read registers in one field in an arbitrary number of register designating fields; and

a register file for outputting data in a plurality of registers having consecutive numbers in accordance with an output from the decoder.

Further, according to another aspect of the invention, there is provided a processor comprising:

a decoder for designating a plurality of write registers in one field in an arbitrary number of register designating fields; and

a register file capable of writing values in a

plurality of registers having consecutive numbers in accordance with an output from the decoder.

The register file includes a plurality of banks, and by reading or writing data from the plurality of banks, the number of ports of reading or writing the data of the respective banks is restricted to be equal to or smaller than the number of the register designating fields, to thereby restrain an increase in a circuit scale caused by reading or writing the data by a number of times larger than the number of the fields.

The number of the plurality of registers having the consecutive number is limited to the n-th power of 2 (n is a natural number), to thereby enable to reduce register selecting circuits.

Further, a data pack operation, which is capable of dealing with a number of the data read from the read registers larger than a number of the data written to the write registers in order to read data from the registers larger in a number than the number of the read register designating fields, is realized without producing invalid portions in the write registers.

Further, a data unpack operation, which is capable of dealing with a number of the data written to the write registers larger than a number of the data read from the read registers such that the data can be written to the registers larger in a number than the number of the write

register designating fields, is realized in parallel without executing data writing a plurality of times.

Further, an operation of outputting the data having a data width wider than a width of input data such that the data can be written to the registers larger in a number than the number of the write register designating fields, is realized without producing an invalid portion in the input data and without mounting a special register having a wider data width.

Further, in order to supply the data respectively to pipes necessitating the data equal to or larger than the number of the register designative fields such as pipes for executing data alignment, a plurality of buses for data are provided between registers and operation pipes in addition to a general purpose bus.

Further, there are provided a plurality of buses for data for writing data to registers between registers and operation pipes for pipes outputting a plurality of results such as data unpack instruction, a permutation instruction, a matrix operational instruction, a multiply accumulate operation instruction and so on.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of CPU (Central Processing Unit) of an SIMD processor according to an example of the invention;

Fig. 2 is a block diagram of a total of the SIMD processor according to the example of the invention;

Fig. 3 is a detailed diagram of a vicinity of a register file portion of the SIMD processor;

Fig. 4 is a diagram showing a truth table of a logic circuit constituting a decode portion;

Fig. 5 is a diagram showing a truth table of a logic circuit constituting a selector for selecting data from banks of Fig. 3;

Fig. 6 is a detailed diagram of individual registers in a bank 203;

Fig. 7 is a diagram showing a truth table of a logic circuit constituting a selector for selecting a location of storing an operation result on a general purpose bus to a register of a bank;

Fig. 8 is a diagram showing a truth table of a logic circuit for constituting a selector for selecting a location of storing an operation result on a bus 210 to a register of a bank;

Fig. 9 is a diagram showing a definition of a data pack instruction;

Fig. 10 is a diagram showing a definition of a data pack instruction including an immediate value;

Fig. 11 is a functional constitution diagram for realizing a data pack instruction;

Fig. 12 is a diagram showing a definition of a data

unpack instruction;

Fig. 13 is a diagram showing a definition of an unpack instruction with regard to 8 bit data;

Fig. 14 is a constitution diagram for realizing a data unpack instruction;

Fig. 15 is a diagram showing a definition of a permutation instruction;

Fig. 16 is a specific explanatory diagram of a permutation instruction;

Fig. 17 is a functional constitution diagram for realizing a permutation instruction;

Fig. 18 is a diagram showing a definition of a permutation instruction;

Fig. 19 is a diagram showing other definition of a permutation instruction;

Fig. 20 is a diagram showing a specific example of using a permutation instruction;

Fig. 21 is a functional constitution diagram for realizing Fig. 20;

Fig. 22 is a diagram showing a definition of a multiply accumulate instruction;

Fig. 23 is a specific explanatory diagram of a multiply accumulate instruction;

Fig. 24 is a functional constitution diagram for realizing Fig. 23;

Figs. 25A and 25B are diagrams showing a program

example indicating an effect of introducing a data pack instruction;

Figs. 26A and 26B are diagrams showing a program example indicating an effect of introducing a data unpack instruction; and

Figs. 27A and 27B are diagrams showing a program example indicating an effect of introducing a permutation instruction.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A detailed explanation will be given of embodiments of the invention in reference to the drawings as follows. Further, in all of the drawings for explaining the embodiments, portions having the same functions are attached with the same notations and repeated explanation thereof will be omitted.

Fig. 1 shows an outline of a block diagram of CPU of an SIMD processor according to an embodiment of the invention. Incidentally, the CPU block portion indicates a portion of CPU 200 in the layout of a microcomputer shown in Fig. 2. Further, in Fig. 2, notation FPU designates a floating-point operation unit, notation CCN designates a cache controller, notation BSC designates a bus state controller, notation TLB designates a translation look-aside buffer and the layout thereof is constructed by a well-known constitution.

The SIMD processor shown in Fig. 1 adopts a 64 bit RISC (Reduced Instruction Set Computer) architecture as an example and is provided with three operands for executing 32 bit fixed length instruction.

According to the example of Fig. 1, an SIMD processor having following five operation pipes is assumed: data alignment (ALN), multiply (MUL), integer operation (INT), load/store (LD/ST) and branch (BRA).

Fig. 3 shows, in details, a portion of making access from an instruction decoder portion to a register file 213 in the block diagram of CPU of the SIMD processor, described above.

A register number of any of R0 through R63 is designated to a portion m which is one of register designating fields of an instruction code 201 and the code is decoded by a decoder 202 to provide direct access to respective registers.

The decoder 202 is constituted by a logic circuit satisfying a truth table shown in Fig. 4. As is apparent also from Fig. 4, among 64 outputs, 4 pieces become high and 4 consecutive registers can be designated at one time. Data read from respective bank is outputted to a selector 204 and a selector 205. The selector 205 is supplied with lower 2 bits of the operand code m as a control signal for determining data to be outputted to a general purpose bus 206 for inputting.

Also the selector 204 is similarly supplied with data from respective banks. Lower 2 bits of the operand code m that is the same as that of the selector 205 are used as a control signal for outputting remaining data excluding the data to be outputted to the general purpose bus.

Fig. 5 shows a truth table showing a behavior of outputs of the selectors 204 and 205. Notations a and b in the table signify the lower 2 bits of the operand code m, described above, output of the selector 205 corresponds to "To 206" and output of the selector 204 corresponds to "To 307". Further, notation "X" of X0 through X3 expressed as output values of the "To 206" column and the "To 307" column of the truth table, represents BNK (that is, bank).

The outputs from the selectors 204 and 205 are exclusive to each other for respective bank and the selectors 204 and 205 are constituted by a logic circuit satisfying the truth table shown in Fig. 5.

With regard to the respective bank, Fig. 6 shows a detailed diagram of the bank 3 (corresponding to 203 in Fig. 3) and respective registers present in the bank 3 (203) are constituted by standard registers of 2 read·1 write for 1 bit in this example.

In Fig. 3, 16 registers of R0, R4,...,R4n are stored in the bank 0, 16 pieces of R1, R5,...,R4n+1 are stored in the bank 1, 16 pieces of R2, R6,..., R4n+2 are stored in

the bank 2, and 16 pieces of R3, R7,...,R4n+3 are stored in the bank 3.

As shown by Fig. 1, results of operation executed by respective operation pipes are outputted to a general purpose bus 207 for outputting. Further, according to the example, there are four outputs from data alignment position pipe and the multiply pipe and therefore, one of the four outputs is outputted to the general purpose bus 207 and remaining three outputs are outputted to buses 210. Data on the buses 210 is inputted to a selector 208.

The selector 208 is the selector of three inputs and 4 outputs and is constituted by a logic circuit satisfying a truth table shown in Fig. 7.

Notations a and b in Fig. 7 designate control signals, notations X, Y and Z of input designate input values, and output designates outputs to the banks 0, 1, 2 and 3 from left.

A control signal inputted to the selector 208 uses lower 2 bits of a code indicated by a destination register designating field (operand code d). Further, data outputted to the general purpose bus 207 is inputted to a selector 209, and is selected and outputted to a bank in which a register to be stored is present.

The selector 209 uses the lower 2 bits of the code (operand code d) indicated by the destination register designating field similarly to the selector 208, and is

constituted by a logic circuit satisfying a truth table shown in Fig. 8.

Notations a and b in Fig. 8 designate values of the lower 2 bits of the destination register designating field (operand code d), notation X designates an input value, and output designates outputs to the banks 0, 1, 2 and 3 from left.

Outputs from the selector 208 and the selector 209 are exclusive to each other for respective bank and when an output of one of the selectors is data, an output from other of the selectors becomes 0. Therefore, the outputs from the selectors 208 and 209 are written to registers in the banks by calculating logical sums for respective banks.

The above-described is the explanation of the SIMD processor capable of designating a plurality of registers.

Next, an explanation will be given of SIMD operation instructions defined by utilizing the characteristic of the SIMD processor.

Generally, according to the SIMD processor, when data in registers are aligned in orders capable of immediately operating the data, the parallelism can maximally be achieved; however, in many cases, it is necessary to permute the data and thereafter execute principal operation. Therefore, a reduction in a number of permutation cycles as less as possible leads to promotion of the function of the SIMD processor.

Data pack instruction as shown by Fig. 9 and 10 is defined by utilizing the characteristic of the invention.

Fig. 9 shows a case in which a shift amount is present in the register. Fig. 10 shows a case in which a shift amount is present in an instruction code as an immediate value.

Fig. 9 shows an operation code for reading a shift amount (Rn) of the data pack instruction from registers, subjecting data in a group of registers designated by notation Rm to a shift processing, and thereafter executing pack operation, and Fig. 10 shows an operation code in the case in which a shift amount (s) is an immediate value in contrast to Fig. 9.

As is known from an explanation of operation in Fig. 9, the instruction is used for packing four pieces of in-register data into a piece of data and storing the data.

A circuit constitution for realizing the instruction is as shown by Fig. 11.

Four pieces of in-register data designated by a register designating field 1 (operand code m) are transmitted to the general purpose bus 206 and buses 307. Further, a shift amount indicated by a register designating field 2 (operand code n) is taken from a general purpose bus 301. The data and the shift amount are respectively inputted to a barrel shifter 302 capable of shifting with arbitrary bits. A division of a fixed-point can be carried

out by the barrel shifter 302.

It is possible that only lower 16 bits of the shifted results are respectively taken out, outputted to the general purpose bus 207 and packed to a single register as 64 bits width data.

Next, data unpack instruction as shown by Fig. 12 and Fig. 13 is defined by utilizing the characteristic of the invention. The data unpack instruction is an instruction for dividing single in-register SIMD data into data of a plurality of registers and storing the data. The definition is carried out in consideration of a case of dealing with 16 bit data in Fig. 12 and a case of dealing with 8 bit data in Fig. 13.

Fig. 12 shows an operation code for dividing data of 64 bits to that of respective 16 bits, subjecting divided data to sign extend of 64 bits, and writing the data to a write register. In Fig. 13, in contrast to Fig. 12, 8 bit data is dealt with and therefore, the number of write registers to be written is doubled.

A detailed explanation will be given of a circuit constitution for realizing the instruction in reference to Fig. 14.

In-register SIMD data designated by the register designative field 1 (operand code m) is transmitted via the general purpose bus 206.

In Fig. 14, there is shown a case for dividing the

in-register SIMD data into 16 bit data. Data transmitted
via the general purpose bus 206 is divided into respective
16 bits, respectively inputted to separate sign extenders
400, subjected to sign extended into 64 bit data, and
outputted to the buses 207 and 210. The data unpack
function can be realized by the above-described operation.

Further, a permutation instruction as shown by Fig.
15 is defined by utilizing characteristic of the invention.
In Fig. 15, there is shown an operation code for reading
four pieces of designated in-register SIMD data, and
storing the SIMD data subjected to designated permutation
for respective elements and permuted to designated
registers.

Generally, "permutation instruction" is effective in
transposition or rotation of a matrix, butterfly operation
of FFT (fast Fourier transform) or the like and a specific
operation is shown in Fig. 16.

According to a conventional permutation instruction,
when permutation operation is carried out with regard to,
for example, two pieces of SIMD data of 16 bit x 4 pieces,
two pieces of 64 bit width registers for storing result are
needed; however, only one piece of a destination register
can be designated and accordingly, separate instructions
are prepared for an upper bit portion and a lower bit
portion of permutation result.

According to the permutation instruction by the

definition, using the characteristic of capable of designating a plurality of registers in one piece of register designating field, with regard to four pieces of source data as shown in Fig. 16, permutation operation of the upper bit portion and the lower bit portion can be carried out at one time and two set portions can simultaneously be operated.

Fig. 17 shows a specific functional constitution diagram. Two pairs of pieces of source data transmitted via the general purpose bus 206 and the buses 307, are respectively divided into those of 16 bit widths and subjected to permutation operation, and then the result is outputted to the buses 207 and 210 to be written to registers.

Further, in the case of complex number data, real numbers and imaginary numbers are frequently present alternately in loaded data and it is frequently necessary to constitute data rows of only real number data and only imaginary number data in operation.

In that case, according to the SIMD processor of the invention, in this embodiment, 8 pieces of SIMD data can simultaneously be read at maximum and therefore, in the case of 16 bit data, a result of 16 data can be calculated at one time by executing permutation operation among 32 pieces of data.

In order to execute a processing of dealing with

complex number data as described above, instructions as shown by Fig. 18 and Fig. 19 may be defined.

Fig. 18 shows an operation code showing a processing of reading 8 pieces of designated in-register SIMD data and, when respective elements are designated by numerals 1, 2, 3 and 4 by enumerating from a right end, extracting only data elements of 1 and 3, and storing the elements into designated registers. Fig. 19 shows an operation code showing a processing of reading SIMD data, and then extracting only data elements of 2 and 4 and storing the data to designated registers.

Specifically, a processing as shown by Fig. 20 is executed and in this case, data of 63 bit-th through 48 bit-th and data of 31 bit-th through 16 bit-th in respective in-register data are extracted.

When an explanation is given to details of operation by a circuit constitution diagram of Fig. 21, 8 sets 32 pieces of SIMD data transmitted via the general purpose bus 206 and buses of 301, 307 and 501, are respectively extracted only for necessary 16 bit data, and results are outputted to the general purpose bus 207 and buses 210 to thereby realize the operation.

Finally, multiply accumulate operation instruction as shown by Fig. 22 is defined by utilizing the characteristics of the present invention.

In Fig. 22, there is shown an operation code showing

a processing of reading two pieces of designated in-register SIMD source data, further reading in-register data constituting a basis of four pieces of accumulation, calculating the accumulation, and thereafter storing SIMD data to four designated registers.

In general multiplication, a result obtained for bit widths of a multiplier and a multiplicand is provided with a doubled bit width and therefore, according to SIMD type 64 bit data holding four pieces of data of 16 bit width, a register having 128 bit width is needed in order to store operation results. As a real solution, there is frequently adopted a method of storing data effective only in lower 32 bits of 64 bits width register and storing the result to 64 bits by sacrificing a parallelism of SIMD. However, when the accumulation is calculated, there is a possibility of further increasing the bit width of the result and even by the above-described method, operation accuracy is deteriorated.

In DSP (Digital Signal Processor), in the case of multiply accumulate operation of 16 bits x 16 bits, accuracy of operation is devised to be maintained by preparing 40 bit registers for storing.

However, when the characteristic of the invention is utilized, multiply accumulate operation without deteriorating accuracy can be executed without deteriorating the parallelism of SIMD.

Fig. 23 shows a specific explanatory view of multiply accumulate instruction defined in Fig. 22 and Fig. 24 shows a circuit constitution diagram.

SIMD data including four pieces of 16 bit data is transmitted via the general purpose bus 206 and the buses 307. The SIMD data is divided into for each 16 bit data, respectively inputted to multipliers 700 through 703, and outputted to adders 704 through 707. Data of accumulation is transmitted via the general purpose bus 301 and buses 501 and is inputted to the adders 704 through 707. Results of the accumulation calculation are outputted to the general purpose bus 207 for storing and the buses 210. By using such means, the multiply accumulate operation of SIMD data can be realized without deteriorating accuracy and with the parallelism being maintained.

Although a specific explanation has been given of the invention carried out by the inventors based on the embodiments, the invention is not limited thereto but can naturally be modified within a range not deviated from the gist.

For example, although the selector 204 in Fig. 1 is a selector of 3 outputs for 4 inputs, it is no problem to constitute the selector by using a tristate buffer.

In the case of the data pack instruction, in Fig. 11, the instruction is carried out by 4 inputs and one output, any number of inputs is acceptable and a number of outputs

is not restricted.

Further, in the case of the data unpack instruction, a number of division is determined by a type of SIMD data in registers and therefore, the number is not limited to four as in Fig. 14.

A simple explanation will be given of effects achieved by representative aspects of the invention disclosed in the application as follows.

That is, according to the SIMD processor of the invention, high speed formation of in-register data alignment operation obstructing an effect of SIMD operation can be realized and multiply accumulate operation as in DSP can be realized.

Specific effects are enumerated as follows.

(1) By introducing the data pack instruction defined by the invention, scatteredly stored data is summarized and data can be processed efficiently by the SIMD instruction.

Figs. 25A and 25B show the case of the embodiment (pack of four pieces of 16 bit data). Fig. 25A designates a program example of the case of the current state and Fig. 25B designates the case of outputting new data pack instruction.

Notation "pack. W" in Fig. 25B designates mnemonic of pack instruction. By adopting the data pack instruction as in the program example of Figs. 25A and 25B, the number of instruction steps can be reduced by 1/4.

(2) By introducing the data unpack instruction defined by the invention, prompt initialization of registers and division of data can be carried out and even at a portion in which the SIMD processing is not effective, operation efficiency can be prevented from being deteriorated.

Figs. 26A and 26B show the case of the embodiment (unpack four pieces of 16 bit data present in one piece of register into four registers).

Fig. 26A designates a program example of the case of data pack by using the conventional permutation instruction and Fig. 26B designates a program example of the case of adopting new data unpack instruction. As is apparent also from Fig. 26B, by adding new instruction, a number of instruction steps can be reduced to 1/6.

(3) By introducing a permutation instruction defined by the invention, permutation of in-register SIMD data can be executed at high speed, and data can be supplied to SIMD processing such as multiple accumulate operation without delay.

Figs. 27A and 27B show the case of the embodiment. Fig. 27A designates a program example in the case of using the conventional permutation instruction and Fig. 27B designates a program example in the case of adopting new permutation instruction. As is apparent from Fig. 27B, the number of instruction steps can be reduced to 1/8.